Technical Disclosure 08/13/12
Jürgen Radzuweit
mailto:juergen@radzuweit.eu

Method to deploy non-standarized software packages in heterogenious environments with make

Managing more than one environments with interconnected applications from different vendors it is difficult to maintain a common standard of how the software packages should be delivered by the vendor. Therefore it is difficult to maintain a common installation routine for all applications. To reduce the manual tasks and get a minimum of standardized procedure, even if there is no deployment tool available (for various reasons), using the build tool make can help. One prerequisite is to have all installable parts on a central machine available.

The advantage of this method is that

a) only available tools, commands, programs are used. No need to install expensive tools or ask a system administrator (aka root) to install things.

b) make is only one binary has a small footprint and is practically available on every platform

c) reduces manual work substantially and increases the quality of configuration tasks

d) simple syntax, easy to learn

This technical disclosure describes a method to deploy software packages of various kinds to one or more environments containing different versions of different operating systems by using make as a central deployment tool. Using only POSIX compliant commands, each variant of make (even Microsoft's *nmake*) is able to behave in the same manner.

This method defines at least three layers of an application to be handled *system*, *subsystem* and *package*. A *system* defines the current version of an application with all *subsystems*, *components* and modules. A *subsystem* is part of a system and contains components and modules and has a version. Software is delivered as parts of the *system* or *subsystem* in various formats we define a *package* as the smallest unit here (this can be extended if required, the method is not bound to the mentioned number of layers). The makefile on each layer has a specific scope and „responsibility“.

The makefile for a system is a master and will be deployed to the master target machine. At the initial state this must be done manual at least once. After that the makefile should have a target to synchronize itself with the central storage. The master makefile calls all other makefiles which are are defined for the other layers as there are subsystem and package. The system makefile “knows” about any child machines (e.g. cluster machines) and deploys itself to them. Also the current environment is known by the makefile and can be controlled by a commandline parameter. The system makefile checks if there are newer packages for a subsystem areb on the central store available and if yes gets them and runs the installation target. This target calls the makefile for each subsystem.

The subsystem makefile knows which packages are available for this version of the subsystem and calls each package makefile with the appropriate parameter.

The package makefile handles the package specific installation, houskeeping and backup tasks etc.

A common set of targets makes sure that all makefiles have a common “understanding” about the application, the version of the application and its subsystem, the packages related to the version of

application/subsystem, where to install how to configure a given environment, which housekeeping has to executed, where to store backup, how to roll back a installtion to a previous version.